Final Report

M. Furkan Atasoy
Bogazici University

Zehranaz Canfes
Bogazici University

Abstract

Supercomputer data is hard to understand and analyze due to the high number of nodes and features. Therefore an automated way should be created to detect the anomalies occurring in the supercomputer. The main objective of the project is to create a Deep Learning model to detect anomalies. For this purpose, Autoencoders are used. However, Autoencoder itself does not provide sufficiently good results due to the real data which is noisy and has unexpected behavior. Therefore, new architectures are introduced that are based on Autoencoder but use a statistical method or a Classifier to improve the results.

1 Introduction

Supercomputers are used for computationally intensive tasks in various fields. They are designed to perform as efficiently as possible. However, because of many different reasons, they may break and stop working. Therefore, it is important to detect unwanted states of the computing nodes to be able to avoid them.

In this project, the Marconi100 supercomputer is used. Marconi100 is a new accelerated GPU cluster of CINECA based on IBM Power9 Architecture. In the last several months, data is collected from these computing nodes and labeled. The labeled data contains information on the states of the computing nodes and shows at which states the computing nodes stopped working.

The aim of this project is to analyze the states and features of the computing nodes and to detect anomalous behavior in a semi-supervised fashion. For this purpose, two approaches are proposed. The proposed approaches use the Neural Network Architecture called Autoencoder to detect anomalies and they further improve the results with the additional methods.

2 Preliminary Analysis

The preliminary analysis of the data contains a visualization of the data and calculating statistics to have a better understanding of the underlying structure of the data. Anomalies are assumed to be labeled with 2.

2.1 Plot of the Labels Versus Time

The overall arrivals of anomalous and normal data in time can be seen as follows for node `r207n02`:

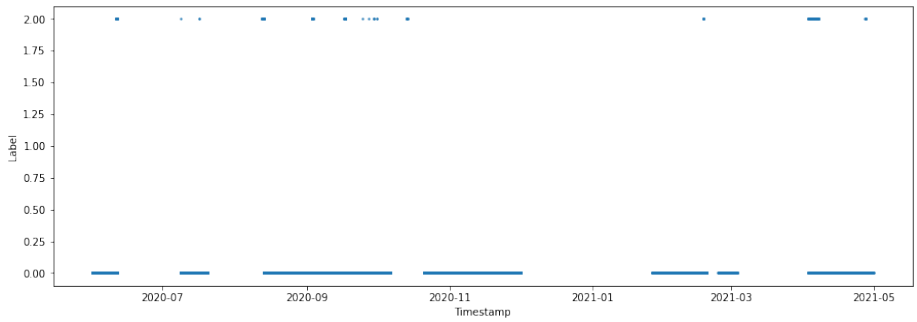


Figure 1: Labels Versus Time for Node `r207n02`

2.2 Ratio of the Anomalies to All Data

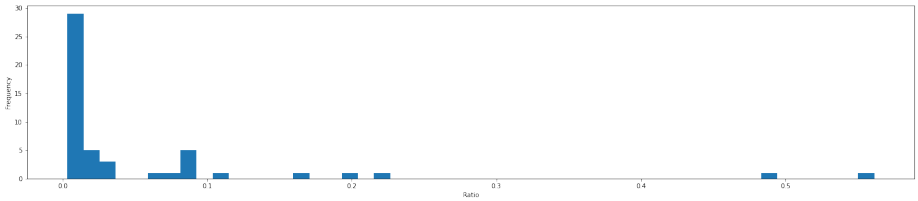


Figure 2: Histogram of the Anomaly Ratios for 50 Nodes

Mean value of the ratios for 50 nodes is 0.05445090412992164.

2.3 Interarrival Distribution and Autocorrelation of the Anomalies

Consecutive interarrivals of anomalies are taken to be one anomaly to observe the interarrivals.

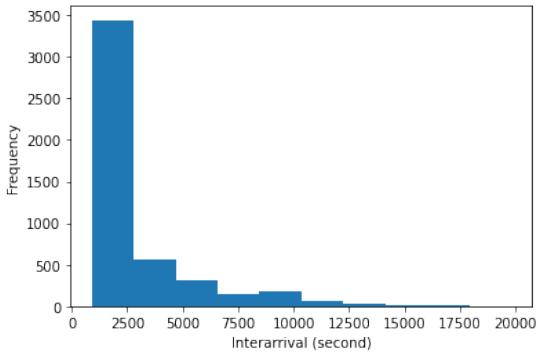


Figure 3: Histogram of the Anomaly Interarrivals for 50 nodes

The interarrivals of anomalies are exponentially distributed according to the results of *Kolmogorov–Smirnov,* and *Chi-Squared* test.

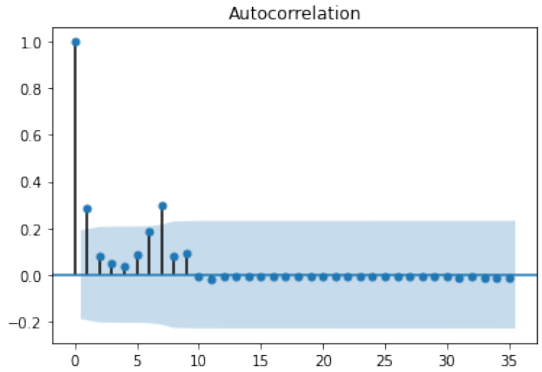


Figure 4: The autocorrelation of the interarrivals of anomalies for node `r207n02`

The autocorrelation is very close to zero which shows that the interarrivals are independent. This result is as desired.

2.4 Heatmap of the Normal and Anomaly Data

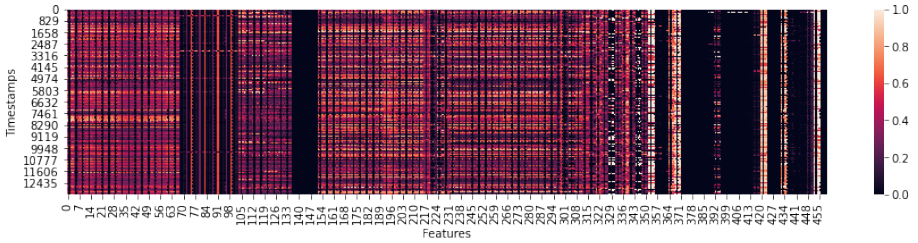


Figure 5: Heatmap of the normal data for node `r207n02`

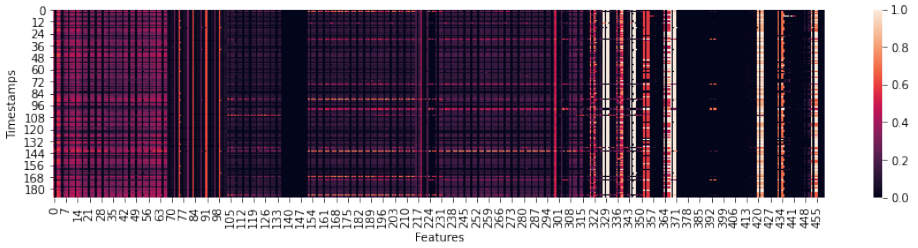


Figure 6: Heatmap of the anomaly data for node `r207n02`

3 Pre-Processing

The collected data is pre-processed to be compatible with the model and also to obtain better results. There are four steps in the pre-processing:

1. The data of April 2021 is filtered from the dataset because it contains unstable and wrong labeling.
2. The data is normalized with MIN-MAX Scaler to be transformed into the range [0, 1] in order to make the training step more effective.
3. The data is shuffled to obtain the same distribution for training, validation, and test dataset.
4. The data is split into three datasets which are training, validation, and test. The training dataset doesn't contain any anomaly because the Autoencoder model should be trained with only normal data. Validation and test datasets have the same amounts of anomalies.

4 Proposed Architectures

To detect anomalies, two different approaches are used:

1. *Autoencoder with post-filtering* where features of the reconstruction losses of Autoencoder are filtered by looking to absolute differences between anomaly and normal reconstruction losses.
2. *Autoencoder with binary classifier* where the latent dimension of the Autoencoder is sent to a binary classifier neural network to find the anomalies in the data.

4.1 Autoencoder with Post-Filtering Model

This proposed architecture starts with an Autoencoder model and is improved with the post-filtering method. The autoencoder model consists of 1 hidden layer and a 330-sized latent dimension.

The proposed architecture is a semi-supervised approach. Because it trains the Autoencoder without labeling and it sets the threshold and applies the post-filtering only with validation dataset (10% training dataset) which contains labeled data.

Post-Filtering is applied to the Autoencoder model by looking at the reconstruction losses of normal data points and anomalies. The features that have the least absolute differences between anomalies and normal data points are filtered. In this proposed architecture, 22 features are chosen at the end. The hidden layer count, latent dimension size, and post-filter count are hyper-parameters that are determined by applying Bayesian Hyperparameter Optimization on 46 nodes. Post-Filtering improves the results since it helps to filter the similar features in anomalies and normal data points so that Autoencoder can distinguish between anomalies and normal data points better.

Autoencoder is compiled with the 'adam' optimizer and the 'mean squared error' loss. It is trained in 100 epochs with 512 batch sizes. The detailed architecture of the Autoencoder can be seen in Figure 7.

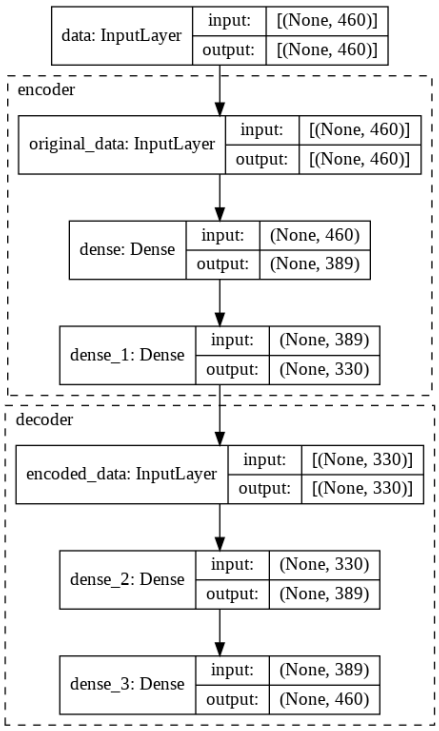


Figure 7: Autoencoder

After training, to improve the F-scores and decrease the number of false positives in the predictions, the post-filtering method is used. This method uses a validation dataset and calculates the mean values of the reconstruction losses both for anomaly and normal data for each feature. It then takes the absolute differences of the mean values and chooses 22 features that have the most absolute differences. Finally, it calculates the new reconstruction losses only by looking at the chosen features.

After post-filtering, the threshold for reconstruction losses is set by using the validation dataset. The threshold for anomalies is calculated by finding the one that gives the best results in terms of f-scores. After that, the threshold is tested on the test dataset and the results are achieved.

The numerical results of Autoencoder with Post-Filtering architecture can be found in *5. Experimental Analysis.*

4.2 Autoencoder with Classifier Model

This proposed architecture starts with an Autoencoder model to calculate the latent features of the data. Then, it uses a binary classifier neural network that takes latent features as input to make a prediction. The Autoencoder model

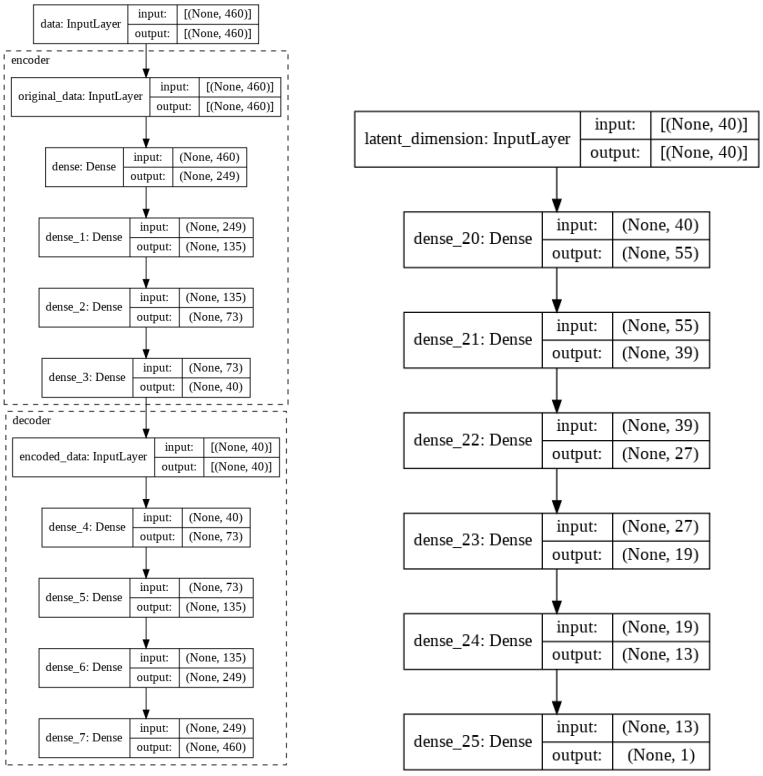
consists of 3 hidden layers and a 40-sized latent dimension, and the Classifier model contains 5 hidden layers which are sized as [55, 39, 27, 19, 13].

The proposed architecture is a semi-supervised approach. Because it trains the Autoencoder without labeling and trains the Classifier only with validation dataset (10% training dataset) which contains labeled data.

The hidden layer count and latent dimension size are the hyper-parameters of the Autoencoder model. The hidden layer count and sizes of the hidden dimensions are the hyper-parameters of the Classifier model. These hyper-parameters are determined by applying *Bayesian Hyperparameter Optimization* on 46 nodes.

Autoencoder is compiled with the 'adam' optimizer and the 'mean squared error' loss. It is trained in 100 epochs with 512 batch sizes. Classifier is compiled with the 'adam' optimizer and the 'binary cross entropy' loss. It is trained in 1000 epochs with 64 batch sizes.

The detailed architectures of the Autoencoder and Classifier can be seen in Figure 8.



(a) Autoencoder (b) Classifier

Figure 8: Autoencoder with Classifier

Because the Classifier is trained only with the validation dataset which contains few amounts of anomaly data, the data oversampling method is used to balance the size of anomalous data with the size of normal data. The oversampling rate is 0.25 for the nodes which have the anomaly ratio less than 7%; 0.5 for the nodes which have the anomaly ratio between 7% and 25%; 1.0 for the nodes which have the anomaly ratio larger than 25%. *Adaptive Synthetic Sampling* (ADASYN) technique is used to oversample the data.

After the oversampling, the Classifier model is trained with the new balanced data. The numerical results of Autoencoder with Classifier architecture can be found in *5. Experimental Analysis.*

5 Experimental Analysis

The metrics for the evaluation are F-scores ($F_1, F_2, F_{0.5}$), accuracy, recall, precision, and confusion matrix. Our analysis mainly focuses on F_2 score which focuses on minimizing the false negatives and is given by:

$$F_2 = (1 + 2^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(2^2 \cdot \text{precision}) + \text{recall}}.$$

F_1 and $F_{0.5}$ scores are also being collected for the general analysis.

In the calculation of the confusion matrix and F-scores, two different labelings are used. One labeling is the original labeling of the data, the other one is the modified version of the original labels which extends the beginning of the anomaly sequences one hour to solve possible wrong-labeling (caused by late detection of the anomaly sequences while labeling). Then, original labels are used to calculate True Negatives and False Negatives. Modified labels are used to calculate True Positives and False Positives. In this way, if an anomalous data point is wrongly labeled as a normal data point, it doesn't affect the number of False Positives.

5.1 Results in a Single Node

For the single node analysis, node `r207n02` is chosen.

5.1.1 Autoencoder with Post-Filtering

After the training of the Autoencoder, the architecture continues with predictions on the data. The reconstruction losses of normal and anomalous data in validation dataset are calculated with *'mean squared error'*. These losses are used to calculate the threshold for distinguishing between anomalies and normal data points. The losses can be seen in Figure 9. The calculated threshold for these losses is 0.0067.

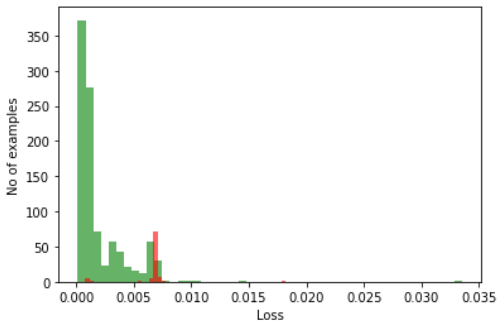


Figure 9: Validation losses in Node `r207n02` before post-filtering (red: anomaly, green: normal)

The model is evaluated on the test dataset and the whole dataset. The metrics collected before the post-filtering can be found in Table 1.

Type	Accuracy	Precision	Recall	F2 Score	Confusion Matrix
Test Dataset	0.93	0.36	0.91	0.70	[[2398 181] [10 102]]
Whole Dataset	0.94	0.20	0.93	0.54	[[12409 817] [17 209]]

Table 1: Metrics for Autoencoder without Post-Filtering

With post-filtering, the features are filtered down to 22 and the losses are calculated using these 22 features. The threshold is automatically calculated by using the histogram of losses in Figure 10. The calculated threshold for these losses is 0.1376.

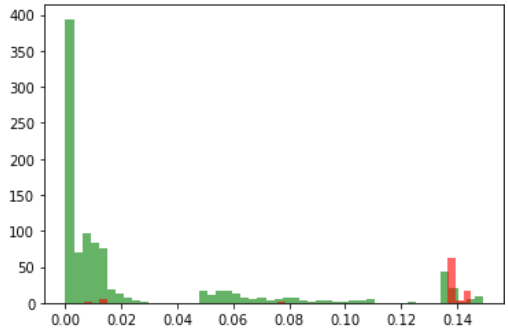


Figure 10: Validation losses in Node `r207n02` after post-filtering (red: anomaly, green: normal)

The model is evaluated on the test dataset and the whole dataset. The metrics collected after the post-filtering can be found in Table 2.

Type	Accuracy	Precision	Recall	F2 Score	Confusion Matrix
Test Dataset	0.97	0.56	0.88	0.79	[[2521 70] [12 88]]
Whole Dataset	0.98	0.37	0.89	0.70	[[12950 303] [20 179]]

Table 2: Metrics for Autoencoder with Post-Filtering

As one can see, post-filtering improves the results by decreasing the false positives. The effect of the post-filtering on all nodes is analyzed in *5.2. Final Comparison.*

5.1.2 Autoencoder with Classifier

After the training of the models, the architecture continues with predictions on the data. The predictions are made according to the result of the output node. If the result of the output node is bigger than 0.5, prediction is an *anomaly* and if the result is less than or equal to 0.5, prediction is *normal*.

The model is evaluated on the test dataset and the whole dataset. The metrics collected for the classifier approach can be found in Table 3.

Type	Accuracy	Precision	Recall	F2 Score	Confusion Matrix
Test Dataset	0.99	0.76	0.89	0.86	[[2574 26] [10 81]]
Whole Dataset	0.99	0.57	0.94	0.83	[[13104 144] [13 191]]

Table 3: Metrics for Autoencoder with Classifier

5.2 Final Comparison

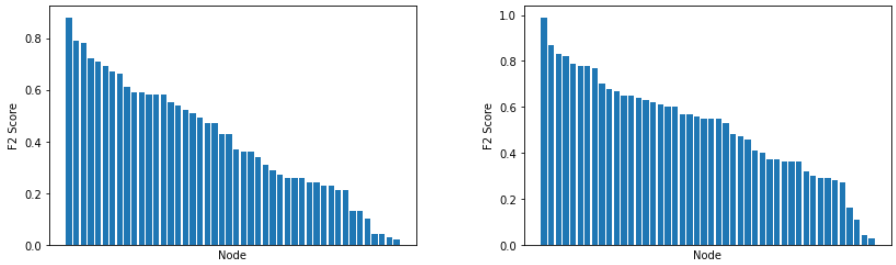
In this section, the proposed architectures will be compared to see the improvements caused by Post-Filtering and Classifier.

5.2.1 Autoencoder without Post-Filtering and Autoencoder with Post-Filtering

To compare the results of the Autoencoder without Post-Filtering and Autoencoder with Post-Filtering they are run on 46 nodes. For each architecture, the average accuracy, precision, recall, and F2-score can be found in Table 4.

Type	Accuracy	Precision	Recall	F2 Score
Autoencoder without Post-Filtering	0.89	0.28	0.61	0.35
Autoencoder with Post-Filtering	0.90	0.34	0.73	0.48

Table 4: Average Results For 46 Nodes



(a) Autoencoder without Post-Filtering (b) Autoencoder with Post-Filtering

Figure 11: Comparison of F2 Scores for 46 Nodes

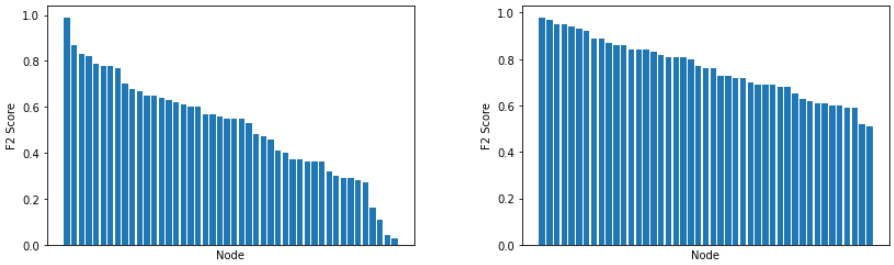
As one can see, with post-filtering the F2 scores are increased but the standard deviation isn't decreased. Also, the post-filtering method increases Accuracy, Precision, and Recall. Thus, it is logical to choose the post-filtering method with the Autoencoder model.

5.2.2 Autoencoder with Post-Filtering and Autoencoder with Classifier

To compare the results of the Autoencoder with Post-Filtering and Autoencoder with Classifier they are run on 46 nodes. For each architecture, the average accuracy, precision, recall, and F2-score can be found in Table 5.

Type	Accuracy	Precision	Recall	F2 Score
Autoencoder with Post-Filtering	0.90	0.34	0.73	0.48
Autoencoder with Classifier	0.98	0.46	0.90	0.73

Table 5: Average Results For 46 Nodes



(a) Autoencoder with Post-Filtering (b) Autoencoder with Classifier

Figure 12: Comparison of F2 Scores for 46 Nodes

As one can see, the standard deviation of F2 scores of 46 Nodes is decreased with the Autoencoder with Classifier Model. Moreover, the F2 scores are increased for every node. It is observed that, architecture with Classifier gives better results in terms of each metric. Precisely, it reduces the false positives and false negatives and thus increases the F2-score.

6 Conclusion

With this report, two architectures to detect anomalies in Marconi100 Supercomputer are proposed. The results of the proposed architectures are promising and it is seen that both architectures give better results compared to the traditional anomaly detection method which only uses an Autoencoder. Even if the results are promising, the proposed architectures can be improved further to decrease the number of False Positives. In order to improve further, one should work with more clean data in which false labeling doesn't occur. Also, the model can be extended to predict the anomalies. However, the sequential order of the data points (time series) shouldn't be changed by shuffling in order to predict the anomalies properly before they come.

Acknowledgements

We would like to thank CINECA for the data they provided.